

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
1	BRS	L1	924	debug adj mode	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:20	
2	BRS	L2	1	1 and storing adj (address near3 (resume near3 instruction))	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:21	
3	BRS	L3	3	1 and (address near3 (resume near3 instruction))	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:21	
4	BRS	L4	0	debug adj mode and rsume adj instruction	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:23	
5	BRS	L5	7	(debug adj mode and resume adj instruction) and stor\$5	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:24	
6	BRS	L6	7	(debug adj mode and resume adj instruction) and stor\$5 near5 address	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:25	
7	BRS	L7	6	6 and fetch\$5 and state	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:26	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
8	BRS	L8	6	6 and fetch\$5 and ((sav\$5 or restor\$5) and (processor near3 state))	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:44	
9	IS&R	L9	1152	(712/228,227,229).CCLS.	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:44	
10	IS&R	L10	2166	(714/30,25,27).CCLS.	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:44	
11	IS&R	L11	0	("09and10").PN.	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:44	
12	BRS	L13	3274	9 or 10	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:45	
13	BRS	L12	44	9 and 10	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:45	
14	BRS	L14	31	12 and debug\$5	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/10/05 09:45	


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

debug and mode and "resume instruction" and register and ad



THE ACM DIGITAL LIBRARY


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

 Terms used **debug** and **mode** and **resume instruction** and **register** and **address** and **debugger** and **PC** and **flip flop** and **multiplexer** and **second**

 Found
11,550
of
143,484

 Sort results
by

relevance


[Save results to a Binder](#)
[Try an Advanced Search](#)
[Try this search in The ACM Guide](#)

 Display
results

expanded form


[Search Tips](#)
☐ Open results in a new window

Results 1 - 20 of 200

 Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

 Relevance scale ☐ ☐ ☐ ☐ ☐

1 [The UT1000 microprogramming simulator: an educational tool](#)

F. Cornett

 June 1989 **ACM SIGARCH Computer Architecture News**, Volume 17 Issue 4

 Full text available: [pdf\(574.05 KB\)](#) Additional Information: [full citation](#), [abstract](#), [index terms](#)

The concepts of microprogramming and firmware are frequently difficult to explain to computer science students. The subject of firmware is normally introduced in a beginning course of the computer science curriculum, usually as one of many terms for which a definition is memorized, but rarely understood. Even in advanced courses in computer organization or architecture, the concept may remain somewhat abstract to the student. Just as the practice of writing and executing programs in high level I ...



2 [Digital test generation and design for testability](#)

John Grason, Andrew W. Nagle

 June 1980 **Proceedings of the 17th conference on Design automation**

 Full text available: [pdf\(1.42 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper is a tutorial intended primarily for individuals just getting started in digital testing. Basic concepts of testing are described, and the steps in the test development process are discussed. A pragmatic approach to test sequence generation is presented, oriented towards ICs interconnected on a board. Finally, design for testability techniques are described, with an emphasis on solving problems that appeared during the test generation discussion.



3 [Testing and Debugging Custom Integrated Circuits](#)

Edward H. Frank, Robert F. Sproull

 December 1981 **ACM Computing Surveys (CSUR)**, Volume 13 Issue 4

 Full text available: [pdf\(2.25 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)


4 [Teaching digital logic design using a tape recorder simulator](#)

R. P. Srivastava


 February 1990 **Proceedings of the 1990 ACM SIGSMALL/PC symposium on Small systems**

 Full text available: [pdf\(658.62 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper describes two implementations of a tape recorder simulator. One is based on hard-wired logic and the other on microcomputer programmed logic approach. Both implementations are compared and evaluated for such points as flexibility, speed, power and space requirements, and cost. The purpose of this paper is to introduce students of digital logic design to the problems of selecting a suitable implementation based on software and




hardware requirements, and system constraints. This is ...

- 5 [A microprogram simulator and compiler for an enhanced version of Tanenbaum's MIC-1 machine](#) 

John L. Donaldson

March 1995 **ACM SIGCSE Bulletin , Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education**, Volume 27 Issue 1

Full text available:  [pdf\(529.29 KB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

- 6 [Reconfigurable computing: architectures and applications: Using reconfigurability to achieve real-time profiling for hardware/software codesign](#) 


Lesley Shannon, Paul Chow

February 2004 **Proceeding of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays**

Full text available:  [pdf\(228.02 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Embedded systems combine a processor with dedicated logic to meet design specifications at a reasonable cost. The attempt to amalgamate two distinct design environments introduces many problems, one being how to partition a single design for the two platforms to achieve the best performance with the least effort. Since the latest FPGA technology allows the integration of soft or hard CPU cores with dedicated logic on a single chip, this presents new opportunities for addressing hardware/software ...

Keywords: FPGA, embedded processor, hardware/software codesign, performance measurement, profiling, soft processor

- 7 [ATLAS/ELA: scan-based software tools for reducing system debug time in a state-of-the-art workstation](#) 

B. I. Dervisoglu, M. A. Keil


June 1989 **Proceedings of the 26th ACM/IEEE conference on Design automation**


Full text available:  [pdf\(434.29 KB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

- 8 [Microprocessor power analysis by labeled simulation](#) 

C. Hsieh, L. Chen, M. Pedram

March 2001 **Proceedings of the conference on Design, automation and test in Europe**

Full text available:  [pdf\(147.20 KB\)](#) Additional Information: [full citation](#), [references](#), [index terms](#)


- 9 [Fast detection of communication patterns in distributed executions](#) 

Thomas Kunz, Michiel F. H. Seuren

November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**


Full text available:  [pdf\(4.21 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

- 10 [THEMIS logic simulator - a mix mode, multi-level, hierarchical, interactive digital circuit simulator](#) 

Mahesh H. Doshi, Roderick B. Sullivan, Donald M. Schuler

June 1984 **Proceedings of the 21st conference on Design automation**

Full text available:  pdf(764.43 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A new logic simulator called THEMISTM Logic Simulator for the design of LSI, VLSI and PCBs is described. THEMIS supports design verification and test development from initial specification in behavioral and RTL languages to analysis of the final layout at the gate and switch level. To allow the simulation of an entire system or check the correctness of a single circuit, the different modeling techniques can be easily intermixed. THEMIS is a highly interactive simulator ...

11 Programmable applications: interpreter meets interface

Michael Eisenberg

April 1995 **ACM SIGCHI Bulletin**, Volume 27 Issue 2


Full text available:  pdf(4.42 MB) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

Current fashion in "user-friendly" software design tends to place an over-reliance on direct manipulation interfaces. To be truly expressive (and thus truly user-friendly), applications need both learnable interfaces and domain-enriched languages that are accessible to the user. This paper discusses some of the design issues that arise in the creation of such *programmable applications*. As an example, we present "SchemePaint," a graphics application that combines a MacPaint-like interface ...

12 MarieSim: The MARIE computer simulator

Linda Null, Julia Lobur

June 2003 **Journal on Educational Resources in Computing (JERIC)**, Volume 3 Issue 2

Full text available:  pdf(340.79 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

MarieSim is a computer architecture simulator based on the MARIE architecture and designed to teach beginning computer organization and architecture. It provides users with interactive tools and simulations to help them deepen their understanding of the operation of a simple computer. Through interaction with MarieSim's graphical environment, students can observe how assembly language statements affect the registers and memory of a computer system. The graphical environment for MarieSim and the a ...

Keywords: Computer architecture simulator, education, introductory architecture

13 Using the Alfa-1 simulated processor for educational purposes

Gabriel A. Wainer, Sergio Daicz, Luis F. De Simoni, Demian Wassermann

December 2001 **Journal on Educational Resources in Computing (JERIC)**, Volume 1 Issue 4

Full text available:  pdf(238.65 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Alfa-1 is a simulated computer designed for computer organization courses. Alfa-1 and its accompanying toolkit allow students to acquire practical insights into developing hardware by extending existing components. The DEVS formalism is used to model individual components and to integrate them into a hierarchy that describes the detailed behavior of different levels of a computer's architecture. We introduce Alfa-1 and the toolkit, show how to extend existing components, and describe how ...

Keywords: DEVS formalism, modeling computer architectures, systems specification

14 A thread-aware debugger with an open interface

Daniel Schulz, Frank Mueller

August 2000 **ACM SIGSOFT Software Engineering Notes , Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis**, Volume 25 Issue 5

Full text available:  pdf(347.13 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


While threads have become an accepted and standardized model for expressing concurrency and exploiting parallelism for the shared-memory model, debugging threads is still poorly supported. This paper identifies challenges in debugging threads and offers solutions to them. The contributions of this paper are threefold. First, an open interface for debugging as an extension to thread implementations is proposed. Second, extensions for thread-aware debugging are identified and implemented with ...

Keywords: active debugging, concurrency, debugging, open interface, threads

15 Migrating a CISC computer family onto RISC via object code translation

Kristy Andrews, Duane Sand

September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems**, Volume 27 Issue 9

Full text available:  [pdf\(1.13 MB\)](#)


Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)



16 Computing curricula 2001

September 2001 **Journal on Educational Resources in Computing (JERIC)**

Full text available:  [pdf\(613.63 KB\)](#)

 [html\(2.78 KB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)



17 iWatcher: Efficient Architectural Support for Software Debugging

June 2004 **Proceedings of the 31st annual international symposium on Computer architecture - Volume 00**

Full text available:  [pdf\(314.11 KB\)](#)

 [Publisher Site](#)

Additional Information: [full citation](#), [abstract](#)

Recent impressive performance improvements in computer architecture have not led to significant gains in ease of debugging. Software debugging often relies on inserting run-time software checks. In many cases, however, it is hard to find the root cause of a bug. Moreover, program execution typically slows down significantly, often by 10-100 times. To address this problem, this paper introduces the Intelligent Watcher (iWatcher), novel architectural support to monitor dynamic execution with minimal overhead ...



18 The micro-architecture of the ECLIPSE® MV/8000: Conception and implementation

Jonathan S. Blau, Charles J. Holland, David L. Keating

November 1980 **Proceedings of the 13th annual workshop on Microprogramming**

Full text available:  [pdf\(622.95 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [index terms](#)


The microcode of the ECLIPSE MV/8000 controls the hardware to emulate an instruction set. In the MV/8000 the micro-architecture is defined and limited by the following constraints: 1) the desire to implement microcode in a limited number of locations; 2) the use of LSI technology; 3) a virtual memory architecture. This paper will attempt to show how each of these factors contributed to the micro-architecture, to describe that architecture, and to relate ...



19 Assembly instruction level reverse execution for debugging

Tankut Akgul, Vincent J. Mooney III

April 2004 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 13 Issue 2

Full text available:  [pdf\(1.13 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Assembly instruction level reverse execution provides a programmer with the ability to return a program to a previous state in its execution history via execution of a "reverse program." The ability to execute a program in reverse is advantageous for shortening software



development time. Conventional techniques for recovering a state rely on saving the state into a record before the state is destroyed. However, state-saving causes significant memory and time overheads during forward execution. Th ...


Keywords: Debugging, reverse code generation, reverse execution

20 A taxonomy of computer program security flaws



Carl E. Landwehr, Alan R. Bull, John P. McDermott, William S. Choi

September 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 3

Full text available:  [pdf\(3.81 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

An organized record of actual flaws can be useful to computer system designers, programmers, analysts, administrators, and users. This survey provides a taxonomy for computer program security flaws, with an Appendix that documents 50 actual security flaws. These flaws have all been described previously in the open literature, but in widely separated places. For those new to the field of computer security, they provide a good introduction to the characteristics of security flaws and how they ...

Keywords: error/defect classification, security flaw, taxonomy

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)